Application Note

TI の湿度センサ：プログラミングおよび統合ガイド

Harry Gill

概要

相対湿度 (RH) センサは、現在の電子システムで重要な役割を果たし、サーバー ルームや産業用オートメーション、電気自動車 (EV) 、スマート インフラに至るまで、幅広いアプリケーションで高精度の環境監視を支援します。これらのセンサは周囲の条件に関するリアルタイムデータを提供することで、敏感なシステムの保護、性能の向上、全体的な信頼性の維持に役立ちます。本アプリケーション ノートには、以下 3 つの製品世代にわたってテキサス インスツルメンツの湿度センサを選択および統合するための実践的なガイダンスが記載されています。HDC1x (第 1 世代) 、HDC2x (第 2 世代) 、HDC3x (第 3 世代および最新世代)。各デバイス ファミリは温度と湿度の測定機能を搭載しており、各シリーズには独自のインターフェイス プロトコルと構成オプションがあります。本アプリケーション ノートで説明する内容は、特定のシステム設計における TI の湿度センサの評価および実装プロセスを簡素化することを目的としています。

注

特に記述のない限り、本アプリケーション ノートが「*デジタル インターフェイス/プロトコル*」への言及は I2C ベースのデジタル通信に厳密に関連しています。本アプリケーション ノートでは、Arduino™ プラットフォームを使用したサンプル コードに焦点を当てています。このプラットフォームは、簡素性と迅速な実装に最適な C ベースのオープンソース プロトタイピング プラットフォームです。その他の C コードは、本資料の最後に記載されているリンクから入手できます。

また、HDC302x や HDC2x など、デバイス名の末尾に「x」の文字が付いている場合、「x」以降の該当するすべてのバリアントに以下の説明が適用されることに注意してください。

HDC1x = HDC1010/HDC1080

HDC2x = HDC2010/HDC2021/HDC2022/HDC2080

HDC3x =HDC3020/HDC3021/HDC3022/HDC3120

目次

商標

1 はじめに

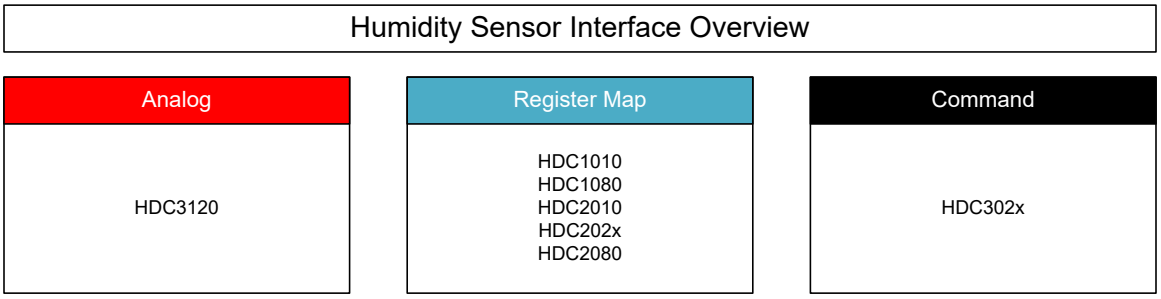

図 1-1. 湿度センサ インターフェイスの概要

テキサス インスツルメンツは、アナログ レシオメトリック通信とデジタル I2C 通信の 2 種類の主なインターフェイスに対応する湿度センサ製品ポートフォリオを提供しています。これらのインターフェイス タイプは、以下の 3 つのグループに分類できます。

1. **デジタル I2C - レジスタ マップ プロトコル**
   - 通信は、レジスタベース (TI の温度センサと同様) です。
   - 特定のレジスタ アドレスへの書き込みと読み取りにより、測定がトリガされ、読み取られます。
   - デバイス:HDC1x および HDC2x (例:HDC1080、HDC2022)
2. **デジタル I2C – コマンドベースのプロトコル**
   - 通信は、コマンド シーケンスに基づいています。
   - ホストが測定を開始するコマンドを送信し、別の読み取りコマンドを使用して結果を取得します。
   - デバイス:HDC302x (例:HDC3020、HDC3022)
3. **アナログ出力 – レシオメトリック電圧**
   - センサは、温度と湿度に比例する電圧信号を出力します。
   - これらは、アナログ システムに直接供給すること、または外部 ADC を使用してデジタル化することができます。
   - デバイス:HDC3120

インターフェイスのタイプごとに、センサ データを読み取るための異なるアプローチが必要です。以下のセクションでは、これらの方法の詳細を説明し、Arduino 対応マイコンのサンプル コードを示します。

2 I2C デジタル インターフェイスの概要

TI は現在、レジスタ マップまたはコマンドベース アクセスという 2 つの I2C インターフェイス形式でデジタル湿度センサを提供しています。以下のセクションでは、各デジタル湿度センサ ファミリからセンサをプログラミングする際の一般的な手順について説明します。

2.1 レジスタ マップ プロトコル

HDC1x および HDC2x のセンサ ファミリは、レジスタ マップベースのデジタル インターフェイスを使用しており、特定のレジスタに書き込むことで、温度と湿度の測定を開始および読み取ります。

通信の概念は似ていますが、2 つのファミリは機能と構成オプションが異なります。

HDC2x シリーズは、以下のようなより高度な機能セットを提供します。

- アラート機能
- データ レディまたは割り込みピンのサポート
- 2 つの測定モード:
  - ユーザー制御のリアルタイム サンプリングのトリガ オンデマンド
  - ローパワー定期サンプリングの自動測定モード (AMM)
- 温度 MSB/LSB (温度 High または Low) および湿度 MSB/LSB (湿度 High または Low) の分割レジスタ

HDC1x シリーズは、より簡単に実装できるように設計されています。トリガ オンデマンドのみがサポートされており、温度と湿度を測定するためにそれぞれ 1 つのレジスタを持つ基本的なレジスタ構造を使用しています。

要約すると、高度な機能、構成可能な測定モード、またはローパワー動作が必要な設計には、HDC1x よりも HDC2x シリーズを推奨します。HDC1x シリーズは、シンプルなデジタル I2C インターフェイスと簡単なコーディング要件で十分なアプリケーション向けに設計されています。

2.1.1 I2C レジスタ マップ プロトコルの簡単な概要

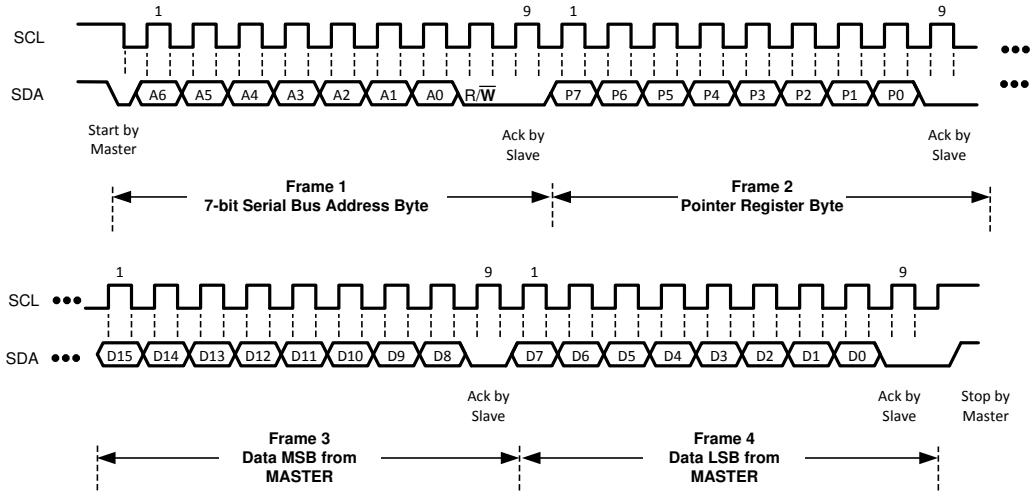

図 2-1. HDC1080 データ フレームの例 (構成レジスタ)

I2C ベースのセンサでは、レジスタ マップはメモリ位置 (レジスタ) の構造化テーブルです。これにより、デバイスの制御方法とアクセス方法が定義されます。各レジスタには固有のアドレスがあり、ホストが読み書きできるため、センサの構成、ステータス ビット、データとの直接的な相互作用が可能です。この整理されたレイアウトにより、設定の調整、測定値の取得、フラグ ビットの監視が容易になります。HDC1x および HDC2x デバイスの場合、データ構造は 図 2-1 のようになります。ここでは、アドレス バイトを送信した後、コントローラはポインタ レジスタ バイトを送信してから、コントローラがセンサからデータを読み取る必要があります。

I2C の詳細については、以下の「I2C の基礎」の資料を参照してください。

2.1.1.1 HDC1x

TI の第 1 世代 HDC1x デバイスはすべて、同じレジスタ マップを共有しています (表 2-1 を参照)。したがって、HDC1x ファミリについては、以下の説明が適用されます。

表 2-1. HDC1x レジスタ マップ

ポインタ	名称	リセット値	説明
0x00	温度	0x0000	温度測定出力
0x01	湿度	0x0000	相対湿度測定出力
0x02	構成	0x1000	HDC1080 の構成およびステータス
0xFB	シリアル ID	デバイスによって異なります	部品のシリアル ID の冒頭の 2 バイト
0xFC	シリアル ID	デバイスによって異なります	部品のシリアル ID の中間の 2 バイト
0xFD	シリアル ID	デバイスによって異なります	デバイスのシリアル ID の末尾のバイト ビット
0xFE	メーカー ID	0x5449	テキサス インスツルメンツの ID
0xFF	デバイス ID	0x1050	デバイスの ID

まず、ユーザーは測定シーケンス (温度、湿度、またはその両方を順番に測定) を定義するために、表 2-2 (0x02) のレジスタに 16 ビットの値を書き込む必要があります。

表 2-2. HDC1x 構成レジスタ (0x02)

名称	ビット			説明
RST	[15]	ソフトウェア リセット ビット	0	通常動作では、このビットは自動的にクリアされます
			1	ソフトウェア リセット
予約済み	[14]	予約済み	0	予約済み、0 にする必要があります
熱	[13]	ヒーター	0	ヒーター ディスエーブル
			1	ヒーター イネーブル
モード	[12]	アクイジションのモード	0	温度または湿度を取得します。
			1	温度と湿度は、まず温度からの順に取得されます。
BTST	[11]	バッテリ ステータス	0	バッテリ電圧 > 2.8V (読み取り専用)
			1	バッテリ電圧 < 2.8V (読み取り専用)
TRES	[10]	温度測定の分解能	0	14 ビット
			1	11 ビット
HRES	[9:8]	湿度測定の分解能	00	14 ビット
			01	11 ビット
			10	8 ビット
予約済み	[7:0]	予約済み	0	予約済み、0 にする必要があります

以下の例では、HDC1x はレジスタに 0x10 (MSB) および 0x00 (LSB) を書き込むことで、温度と湿度の両方を順番に測定するように構成されています。図 2-2 に、設定されるレジスタ ビットを示します。

HDC1x が温度と湿度を順次出力するように設定されている場合、ユーザーは温度レジスタ (0x00) から 4 バイトの読み取りを開始する必要があります。

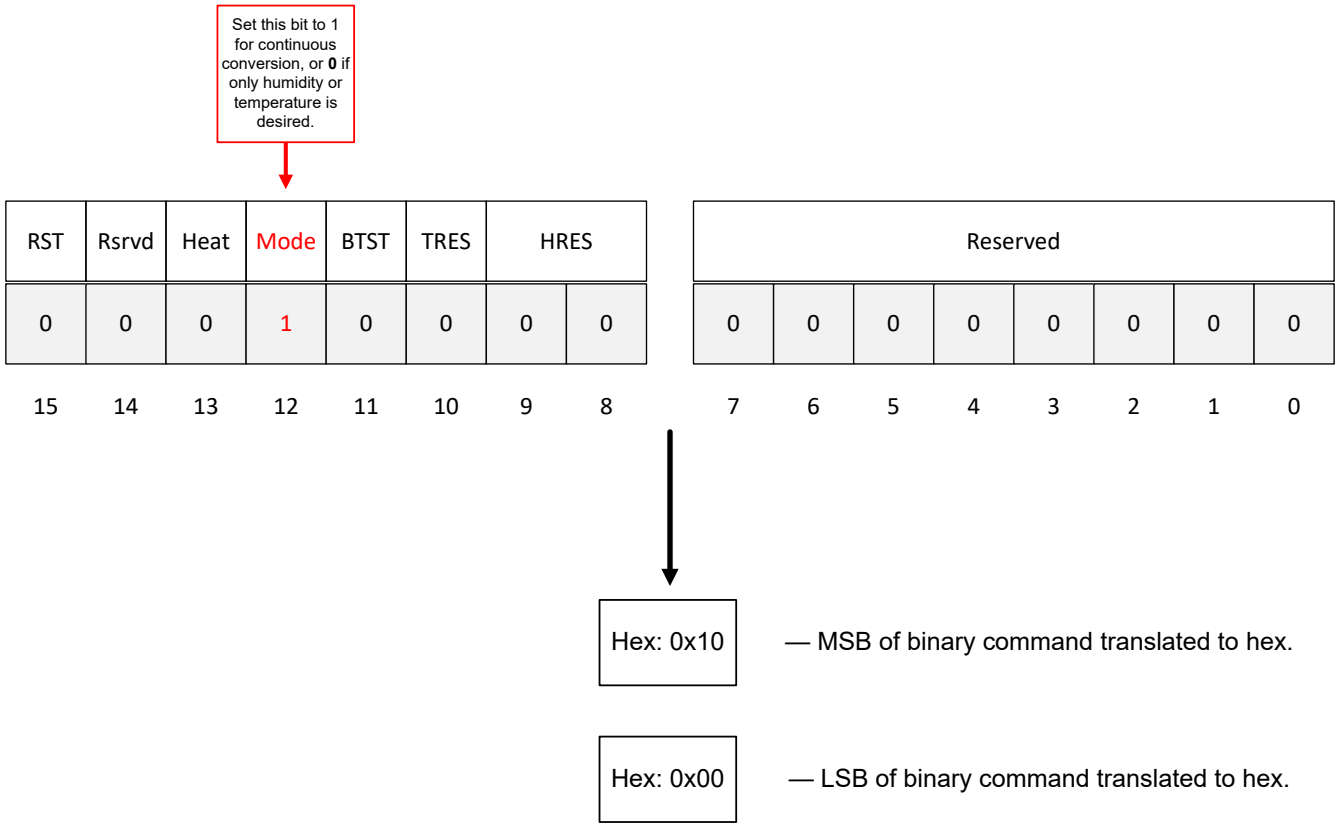

図 2-2. 温度および湿度アクイジションの構成レジスタ ビット

構成を設定するためのコードは以下のとおりです。

```
Wire.beginTransmission(0x40); // initiate communication with HDC1x
Wire.write(0x02); // point to configuration register
Wire.write(0x10); // write 8-bit configuration to config register (MSB)
Wire.write(0x00); // write 8 0s to Reserved bits (LSB)
Wire.endTransmission();
```

次に、デバイス アドレス (0x40) に書き込み、測定プロセスをトリガします。

```
Wire.beginTransmission(0x40); // initiate communication with HDC1x
Wire.write(0x00); // start measurements
Wire.endTransmission();
delay(20); // wait 20ms for conversion to complete.
```

温度と湿度は順番に測定されるため、レジスタ 0x00 から 4 バイトを読み取る必要があります。最初の 2 バイトは温度に対応し、次の 2 バイトは湿度データに対応しています。

```
Wire.requestFrom(0x40, 4); // requesting 4 bytes from device

// once 4 bytes are received, store this in appropriate variables
if (Wire.available() == 4) {

    // stores raw temperature and humidity data
    // reads/stores first byte (MSB), then reads/stores second byte
    // combines each pair of bytes into a 16-bit integer

    uint16_t tempBytes = (Wire.read() << 8) | Wire.read();
    uint16_t humBytes = (Wire.read() << 8) | Wire.read();
}
```

最後に、HDC1x データシートから標準の変換式を適用します。

```
// equation for converting temperature output in Celsius
temp = (tempBytes / 65536.0) * 165.0 - 40.0;

// equation for converting humidity output
hum = (humBytes / 65536.0) * 100.0;
```

この Arduino の例では、測定を順に取得する際に、HDC1x センサから温度および湿度データを読み出し、保存する方法を示します。未加工データを保存すると、HDC1x データシートに記載されている式を使用して、物理的な温度と湿度の値に変換できます。

こちらの TI GitHub リポジトリにある環境センサのセクションで完全な動作サンプルを入手できます。

2.1.1.2 HDC2x

HDC2x ファミリ (HDC2010、HDC2021、HDC2022、HDC2080) でも、HDC1x シリーズと同様、レジスタ マップベースのデジタル インターフェイスを使用します。表 2-3 に示されているように、すべての HDC2x デバイスは共通のレジスタ レイアウトを共有しており、以下の手順はファミリ全体で適用されます。このセクションでは、トリガ オンデマンド (ワンショット) および自動測定 (連続変換) の両方のモードでこれらのデバイスと接続する方法について説明します。

表 2-3. HDC2x レジスタ マップ

ポインタ	名称	リセット値	説明
0x00	TEMPERATURE LOW	0x00	温度 [7:0]
0x01	TEMPERATURE HIGH	0x00	温度 [15:8]
0x02	HUMIDITY LOW	0x00	湿度 [7:0]
0x03	HUMIDITY HIGH	0x00	湿度 [15:8]
0x04	INTERRUPT/DRDY	0x00	データ準備完了および割り込みの構成
0x05	TEMPERATURE MAX	0x00	最大測定温度 (自動測定モードではサポートされていない)
0x06	HUMIDITY MAX	0x00	最大測定湿度 (自動測定モードではサポートされていない)
0x07	INTERRUPT ENABLE	0x00	割り込みイネーブル
0x08	TEMP_OFFSET_ADJUST	0x00	温度オフセットの調整
0x09	HUM_OFFSET_ADJUST	0x00	湿度オフセットの調整
0x0A	TEMP_THR_L	0x00	温度スレッショルド LOW
0x0B	TEMP_THR_H	0xFF	温度スレッショルド HIGH
0x0C	RH_THR_L	0x00	湿度スレッショルド LOW
0x0D	RH_THR_H	0xFF	湿度スレッショルド HIGH
0x0E	RESET&DRDY/INT CONF	0x00	ソフト リセットと割り込み構成
0x0F	MEASUREMENT CONFIGURATION	0x00	測定の構成
0xFC	MANUFACTURER ID LOW	0x49	メーカー ID 下位バイト
0xFD	MANUFACTURER ID HIGH	0x54	メーカー ID 上位バイト
0xFE	DEVICE ID LOW	0xD0	デバイス ID 下位バイト
0xFF	DEVICE ID HIGH	0x07	デバイス ID 上位バイト

注

以下の HDC2x の例では、アドレス 0x40 (ADDR ピンは GND に接続) に構成された HDC2010 を使用していますが、設定に応じてアドレスを簡単に調整するためにグローバル変数を使用できます。

HDC1x シリーズとの主な違いの 1 つは、HDC2x デバイスが各測定の最上位ビットと最下位ビットを保存するために、別の 8 ビット レジスタを使用することです。

- 温度:TEMP_LOW (LSB)、TEMP_HIGH (MSB)
- 湿度:HUM_LOW (LSB)、HUM_HIGH (MSB)

これらのデータ レジスタに加えて、HDC2x シリーズには「**測定構成レジスタ**」が内蔵されており、ユーザーは測定パラメータを定義できます。

測定プロセスは、構成レジスタに書き込むことから開始されます。構成レジスタは以下のような主要な機能を制御します (しかし、これらに限定されるものではありません)。

- ヒーター イネーブル (HEAT_EN)
- 自動測定モード (AMM)
- ソフト リセット (SOFT_RES)

次の手順では、測定パラメータの設定と変換の開始が含まれます。この詳細については、以下のサブセクションで説明します。

2.1.1.2.1 トリガ オンデマンド モードのインターフェイス

この例では、HDC2010 センサが「**トリガ オンデマンド (ワンショット) モード**」に構成され、自動測定モード (AMM) は無効化されています。この手順は、構成レジスタ (0x0E) に書き込み、デバイスをトリガ オンデマンド モードに設定することから始まります。図 2-3 に、構成レジスタの設定を示します。

表 2-4. 構成レジスタ (0x0E)

ビット	フィールド	タイプ	リセット	説明
7	SOFT_RES	R/W	0	0 = 通常動作モード、このビットはセルフクリアされる 1 = ソフト リセット EEPROM の値がリロードされ、レジスタがリセットされる
[6:4]	AMM[2:0]	R/W	000	自動測定モード (AMM) 000 = 無効。I2C により測定開始 001 = 1/120Hz (2 分ごとに 1 サンプル) 010 = 1/60Hz (1 分ごとに 1 サンプル) 011 = 0.1Hz (10 秒ごとに 1 サンプル) 100 = 0.2Hz (5 秒ごとに 1 サンプル) 101 = 1Hz (1 秒ごとに 1 サンプル) 110 = 2Hz (1 秒ごとに 2 サンプル) 111 = 5Hz (1 秒ごとに 5 サンプル)
3	HEAT_EN	R/W	0	0 = ヒータ オフ 1 = ヒータ オン
2	DRDY/INT_EN	R/W	0	DRDY/INT_EN ピン構成 0 = High Z 1 = イネーブル
1	INT_POL	R/W	0	割り込み極性 0 = アクティブ LOW 1 = アクティブ HIGH
0	INT_MODE	R/W	0	割り込みモード 0 = レベル センシティブ 1 = コンパレータ モード

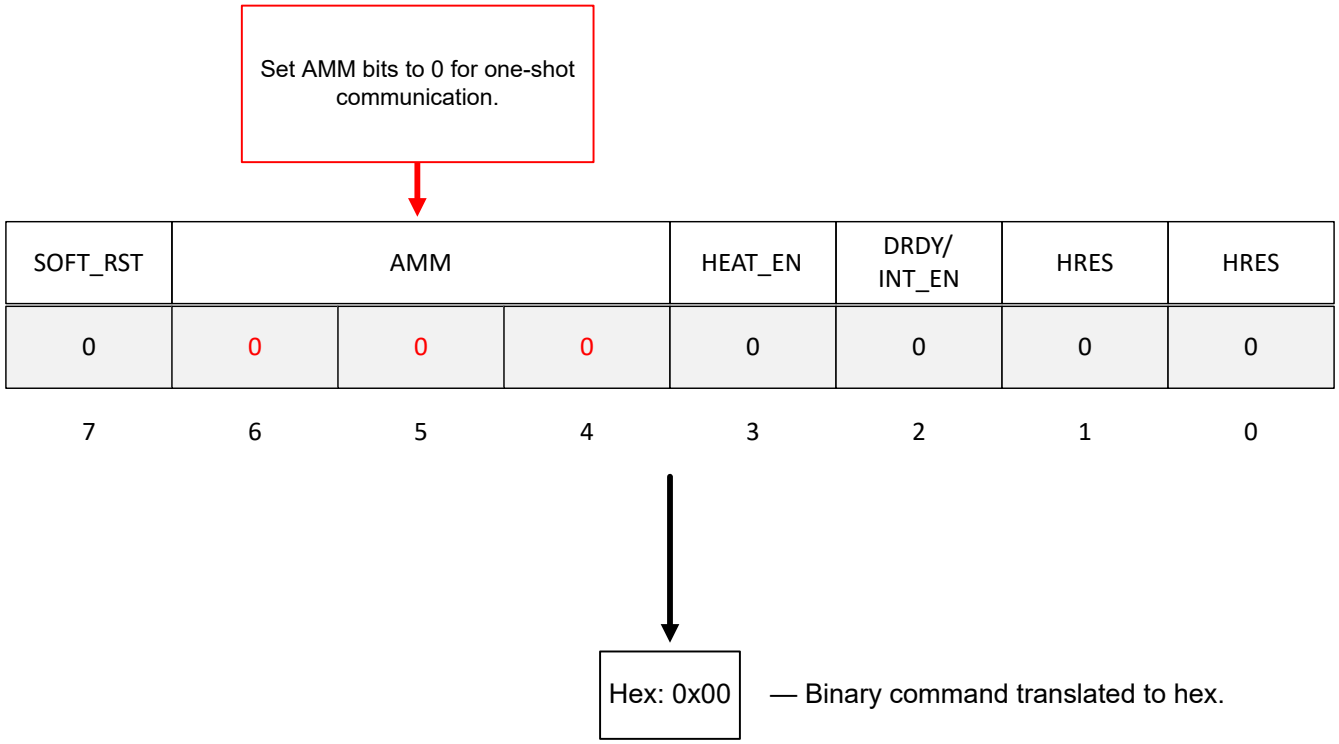

図 2-3. トリガ オンデマンドの構成レジスタ (ワンショット)

```
Wire.beginTransmission(0x40); // initiate communication with HDC2x sensor
Wire.write(0x0E); // write to Config Register
Wire.write(0x00); // configure device to Trigger-On Demand
Wire.endTransmission();
```

図 2-4 に示されているように、次に測定構成レジスタ (0x0F) を構成します。

測定構成レジスタは、以下を定義します。

- 温度と湿度の分解能 (TRES および HRES)
- 測定タイプ (温度のみ、湿度のみ、またはその両方)
- 測定トリガ (MEAS_TRIG)

表 2-5. 測定構成レジスタ (0x0F)

ビット	フィールド	タイプ	リセット	説明
7:6	TRES[1:0]	R/W	00	温度分解能 00:14 ビット 01:11 ビット 10:9 ビット 11:該当なし
5:4	HRES[1:0]	R/W	00	湿度分解能 00:14 ビット 01:11 ビット 10:9 ビット 11:該当なし
3	RES	R/W	0	予約済み
2:1	MEAS_CONF[1:0]	R/W	00	測定構成 00:湿度 + 温度 01:温度のみ 10:該当なし 11:該当なし

表 2-5. 測定構成レジスタ (0x0F) (続き)

ビット	フィールド	タイプ	リセット	説明
0	MEAS_TRIG	R/W	0	測定トリガ 0:反応なし 1:測定開始 測定完了時のセルフクリア ビット

図 2-4 に、測定構成レジスタの図を示します。

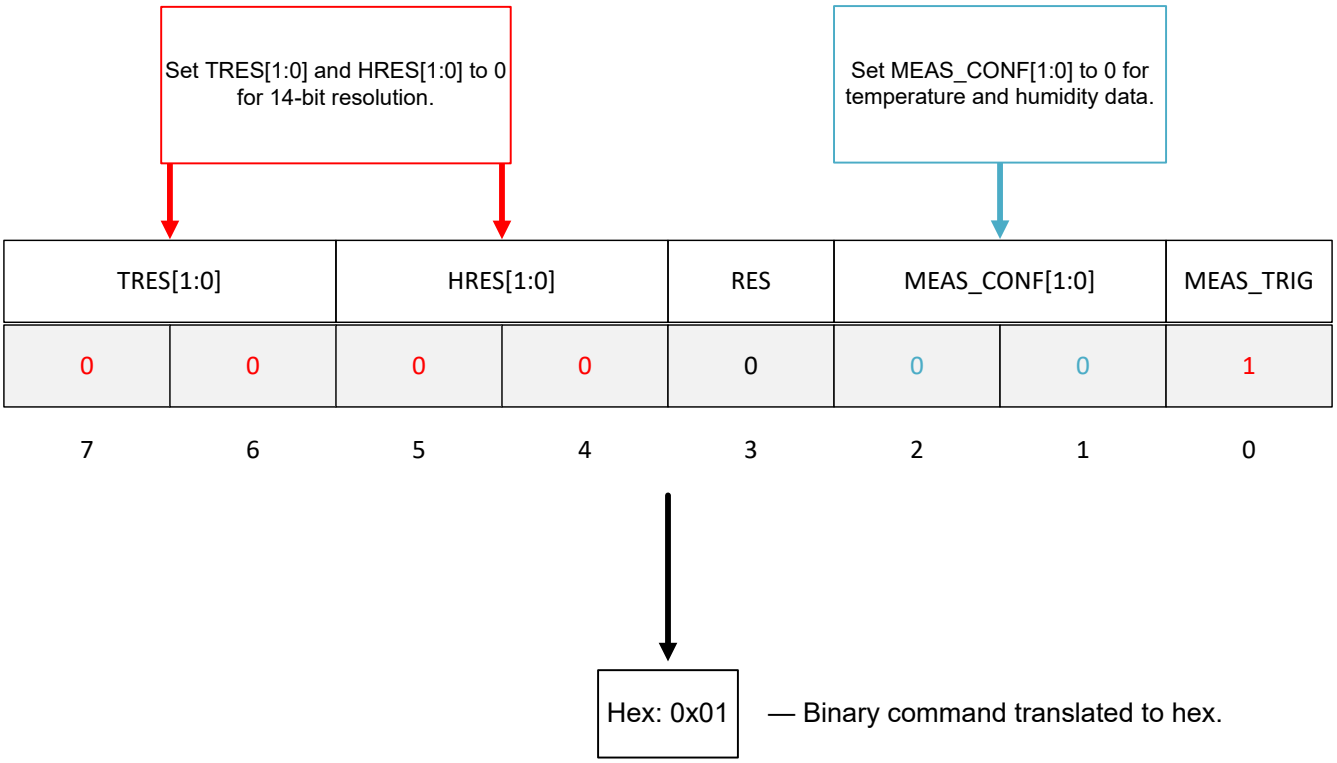

図 2-4. 測定構成レジスタ設定

```
Wire.beginTransmission(0x40); // initiate communication with HDC2x sensor

Wire.write(0x0F); // write to Measurement Config Register

Wire.write(0x01); // set output to 14-bit temperature/humidity data
                  // and trigger measurements

Wire.endTransmission();
```

トリガ オンデマンド モードでは、システムのニーズに応じて、周期的なポーリングのループ内にこのシーケンスを配置できます。

HDC2x の温度と湿度の測定値を読み取るには、ユーザーは 2 つの方法から選択できます。1 つ目の方法は、温度バイトと湿度バイトを TEMP_LOW/HUM_LOW と TEMP_HIGH/HUM_HIGH を使用して個別に読み出し/保存し、MSB ビットと LSB ビットを 1 つの 16 ビット値に結合するか、以下の湿度データ アクイジションのコード に示されているように、1 つの通信フレームの複数バイトのバースト読み取りを行います。

測定データの読み取り

HDC2x は、2 つの 8 ビット レジスタに測定結果を保存します。

- 湿度:HUM_LOW (0x02)、HUM_HIGH (0x03)
- 温度:TEMP_LOW (0x00)、TEMP_HIGH (0x01)

1 つのアプローチは、バイトを別々に読み取って、それらを組み合わせることです。

```
uint16_t getHum() {

    // RH LSB Acquisition

    Wire.beginTransmission(0x40); // start communication with HDC2x
    Wire.write(0x02); // set a pointer for Humidity Low register (0x02)
    Wire.requestFrom(0x02, 2); // request 2 bytes from HDC2x
    uint8_t humLow = Wire.read(); // store Humidity LSB data
    Wire.endTransmission();

    // RH MSB Acquisition

    Wire.beginTransmission(0x40); // start communication with HDC2x
    Wire.write(0x03); // set a pointer for Humidity High register (0x03)
    Wire.requestFrom(0x40, 2); // request 2 bytes from HDC2x
    uint8_t humHigh = Wire.read(); // store Humidity MSB data
    Wire.endTransmission();

    // combine MSB and LSB into 16-bit integer and return value

    return ((uint16_t) humHigh << 8) | humLow;

}
```

しかし、より効率的な方法として、LSB レジスタから両バイトのバースト読み取りを行う手段が挙げられます。

```
uint16_t getHum2() {

    Wire.beginTransmission(0x40); // start communication with HDC2x
    Wire.write(0x02); // set a pointer for Humidity Low register (0x02)
    Wire.endTransmission(false);
    Wire.requestFrom(0x40, 2); // request 2 bytes from HDC2x

    uint8_t lsb = Wire.read(); // read and store LSB
    uint8_t msb = Wire.read(); // read and store MSB

    // adds MSB of data to an empty 16-bit variable
    // shifts 8 bits left, then "or" with LSB for final value

    return ((uint16_t) msb << 8) | lsb;

}
```

このアプローチは、HDC2x の内部ポインタの動作を利用します。HUMIDITY_LOW レジスタから LSB を読み取ると、ポインタは MSB の HUMIDITY_HIGH レジスタに対して自動的にインクリメントします。温度の読み取りにも同じメカニズムが適用されます。

このリンクで、HDC2x のトリガ オンデマンド モードのサンプル コード全体を表示することができます。

2.1.1.2.2 自動測定モード (AMM) を使用したインターフェイス

このセクションでは、HDC2x デバイスを自動測定モード (AMM) で動作するように構成する方法について概説し、トリガ オンデマンド モードと比較した場合の主な違いを説明します。

AMM では、デバイスはユーザー定義のサンプリング周波数で自動的に測定を実行するため、マイコン から手動で測定トリガを行う必要はありません。トリガ オンデマンドでは、各測定を手動で開始する必要がありますが、AMM では周期的な変換を開始するために必要なトリガは 1 つのみです。

この例では、HDC2010 は 5 秒 (0.2Hz) に 1 回サンプリングするように構成されています。これは、適切な設定を構成レジスタ (0x0E) に書き込むことで行います。図 2-5 に、構成レジスタの設定を示します。

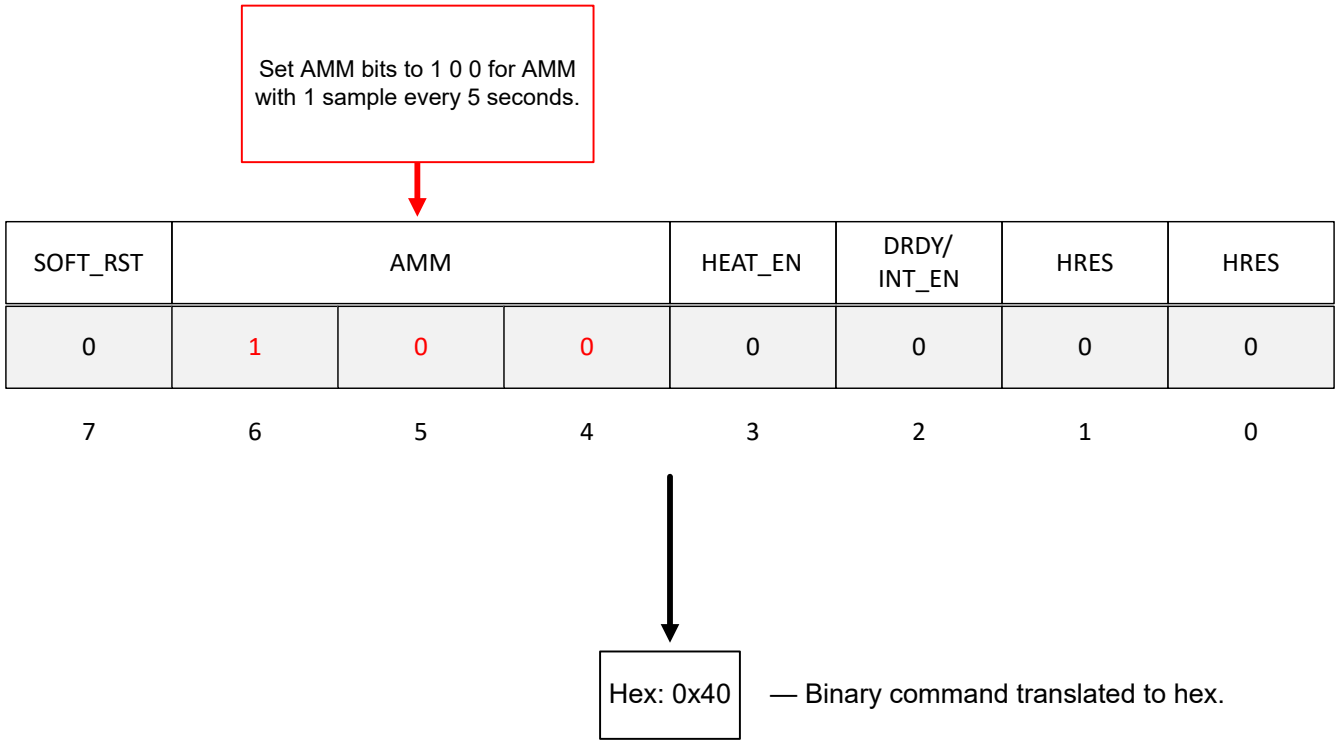

図 2-5. 自動測定モード (AMM) の構成レジスタ

```
// set device to Auto Measurement Mode for 0.2Hz (1 sample/5 seconds)

Wire.beginTransmission(0x40); // start communication with HDC2x

Wire.write(0x0E); // point to register 0x0E (Measurement Config)
Wire.write(0x40); // write value to register

Wire.endTransmission(); // end communication
```

測定構成レジスタは、「トリガ オンデマンド」セクションで説明されている構成と同じものを使用します。

このリンクで、HDC2x の自動測定 モードのサンプル コード全体を表示することができます。

2.2 コマンド プロトコル

2.2.1 HDC302x

HDC302x ファミリにはコマンドベース インターフェイスが導入されており、前世代で使用されていたレジスタ マップベースの方式から脱却しています。HDC302x デバイスは、特定のレジスタへの書き込みの代わりに、適切に定義されたコマンドコードに応答し、測定の開始、設定の構成、データの取得を行います。表 2-6 に、HDC302x デバイスでサポートされているコマンドの例を示します。

このアプローチにより、インターフェイスが簡素化され、特に基本的な測定のみを必要とするアプリケーションで、必要な I2C トランザクションの数が削減されます。

以降のセクションでは、トリガ オンデマンドと自動測定モードの両方で HDC302x と接続する方法を示し、それぞれの場合にコマンドベース通信を実装する方法の例を示します。

表 2-6. HDC302x コマンド表スニペット

16 進コード (MSB)	16 進コード (LSB)	コマンド	コマンドの詳細
24	00	トリガ オンデマンド モード 単一温度 (T) 測定 および相対湿度 (RH) 測定	低消費電力モード 0 (最小限のノイズ)
24	0B		低消費電力モード 1
24	16		低消費電力モード 2
24	FF		低消費電力モード 3 (最小限の電力)
20	32	自動測定モード 2 秒あたり 1 測定。	低消費電力モード 0 (最小限のノイズ)
20	24		低消費電力モード 1
20	2F		低消費電力モード 2
20	FF		低消費電力モード 3 (最小限の電力)
21	30	自動測定モード 1 秒あたり 1 測定。	低消費電力モード 0 (最小限のノイズ)
21	26		低消費電力モード 1
21	2D		低消費電力モード 2
21	FF		低消費電力モード 3 (最小限の電力)
22	36	自動測定モード 1 秒あたり 2 測定。	低消費電力モード 0 (最小限のノイズ)
22	20		低消費電力モード 1
22	2B		低消費電力モード 2
22	FF		低消費電力モード 3 (最小限の電力)
23	34	自動測定モード 1 秒あたり 4 測定。	低消費電力モード 0 (最小限のノイズ)
23	22		低消費電力モード 1
23	29		低消費電力モード 2
23	FF		低消費電力モード 3 (最小限の電力)
27	37	自動測定モード 1 秒あたり 10 測定。	低消費電力モード 0 (最小限のノイズ)
27	21		低消費電力モード 1
27	2A		低消費電力モード 2
27	FF		低消費電力モード 3 (最小限の電力)
2C	06	トリガ オンデマンド モード 単一温度 (T) 測定 および相対湿度 (RH) 測定	低消費電力モード 0 (最小限のノイズ)
2C	0D		低消費電力モード 1
2C	10		低消費電力モード 2

表 2-6. HDC302x コマンド表スニペット (続き)

16 進コード (MSB)	16 進コード (LSB)	コマンド	コマンドの詳細
30	93	自動測定モード	終了してから、トリガ オンデマンド モードに戻ります。
E0	00		T と RH の測定値読み出し (注:RH と T が更新されない場合、データ読み出しはすべて FF になります)
E0	01		RH のみの測定値読み出し
E0	02		最小 T の測定履歴読み出し。
E0	03		最大 T の測定履歴読み出し。
E0	04		最小 RH の測定履歴読み出し。
E0	05		最大 RH の測定履歴読み出し。

2.2.1.1 トリガ オンデマンド モードのインターフェイス (ワンショット)

このセクションでは、高分解能設定を使用して HDC302x デバイスから測定値の構成および読み取りを行う方法を説明します。すべてのサンプル コードは、I2C アドレスのグローバル変数を使用しており、ユーザーのデバイス構成に基づいて簡単に変更できます。

注

この例で使用している HDC302x センサはアドレス 0x44 (ADDR および ADDR1 ピンは GND に接続) に構成されていますが、コード内でグローバル変数を利用できるようになっているため、レイアウトに合わせてアドレスを簡単に調整できます。

測定値を読み取る前に、まずデバイスを構成する必要があります。これは、特定のコマンド シーケンスを HDC302x に送信することで行います。図 2-6 に、構成図を示します。

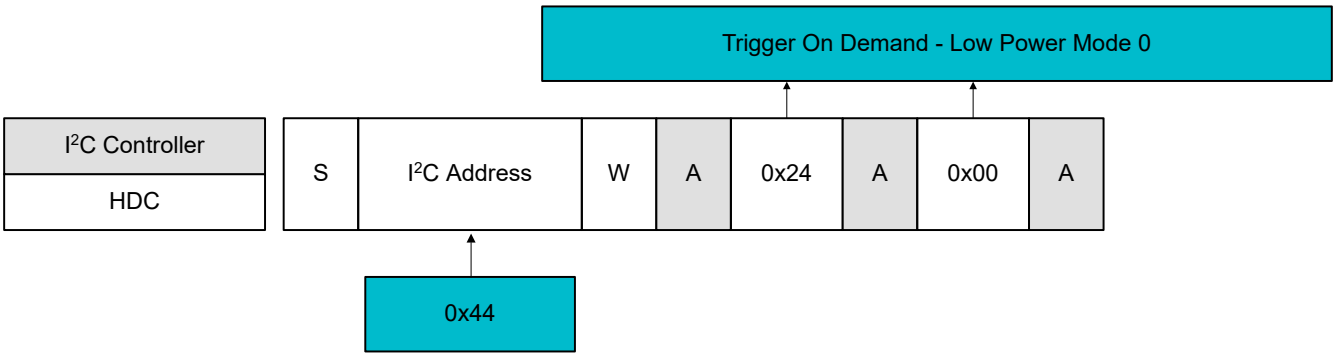

図 2-6. トリガ オンデマンド コマンドの選択

```
Wire.beginTransmission(0x44); // Initiate communication with HDC302x

Wire.write(0x24); // Write Command MSB to device.

Wire.write(0x00); // Write Command LSB to device.

Wire.endTransmission();

delay(25); //wait 25ms before reading
```

読み取りを開始する前に、測定変換が完了したことを確認するための遅延が必要です。この例では、測定構成は最高の分解能と再現性に基づいているため、15ms の遅延を使用しています。エンジニアは、適切な遅延時間を設定する前にデータシートを確認する必要がありますが、15ms の最小遅延を実装するべきです。

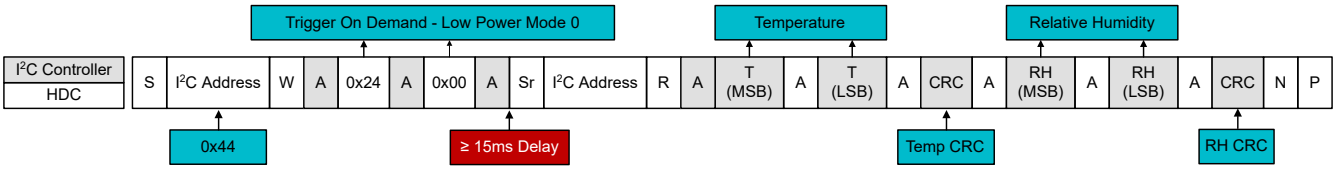

図 2-7. HDC302x トリガ オンデマンド通信構造

```
void loop() {

    float humidity;
    float temp;

    // send device command for highest repeatability
    Wire.beginTransmission(0x44);
    Wire.write(0x24); //send MSB of command
    Wire.write(0x00); //command LSB
    Wire.endTransmission();
```

```
    delay(15); //wait 15ms before reading

    Wire.requestFrom(0x44, 6); //request 6 bytes from HDC device
    Wire.readBytes(HDC_DATA_BUFF, 6); //move 6 data bytes into
buffer

    temp = getTemp(HDC_DATA_BUFF);
    Serial.print("Temp (C): "); // print final temp value
    Serial.println(temp);

    delay(1000); // wait 1 second (optional)

    humidity = getHum(HDC_DATA_BUFF);
    Serial.print("Humidity (RH): "); // print final humidity value
    Serial.print(humidity);
    Serial.println("%");

    delay(1000); // wait 1 second (optional)

}
```

データ変換関数

温度と湿度は、HDC302x データシートに記載されている式を使用して計算します。

```
// function processes raw temperature values and returning final value
float getTemp(uint8_t humBuff[]) {

    float tempConv;
    float celsius;

    TEMP_MSB = humBuff[0] << 8 | humBuff[1]; //shift 8 bits of data in
                                             //first array index to get
                                             //MSB then OR with LSB

    tempConv = (float)(TEMP_MSB); // convert uint8_t temp value
    celsius = ((tempConv / 65535) * 175) - 45; // calculate celcius

    return celsius;

}

// function for processing raw humidity values and returning final value
float getHum(uint8_t humBuff[]) {

    float humConv;
    float humidity;

    HUM_MSB = (humBuff[3] << 8) | humBuff[4]; //shift 8 bits of data in
                                              //first array index to get
                                              //MSB then OR with LSB

    humConv = (float)(HUM_MSB); // convert uint8_t humidity value
    humidity = (humConv / 65535) * 100; // calculate humidity

    return humidity;

}
```

バッファ構造に関する注意事項

6 バイト バッファ HDC_DATA_BUFF には以下が含まれます。

- バイト 0 ～ 1：未加工温度データ (MSB、LSB)
- バイト 2：温度 CRC (オプション)
- バイト 3 ～ 4：未加工湿度データ (MSB、LSB)
- バイト 5：湿度 CRC (オプション)

この例では CRC バイトは使用していませんが、完全性を期すためにバッファに含められており、必要に応じてデータ整合性のチェックに使用できます。

このリンクで、HDC302x のトリガ オンデマンド モードのサンプル コード全体を表示することができます。

2.2.1.2 自動測定モード (AMM) のインターフェイス

このセクションでは、HDC302x を AMM に設定する方法と、トリガー オンデマンド モードとの主な違いについて説明します。

自動測定モードとトリガ オンデマンドの主な違いは、ユーザーが HDC302x センサから一定間隔でデータを読み取ることを想定している場合、プログラム可能な出力間隔を備えた自動測定モードの方が適していることです。図 2-8 に、HDC302x をプログラムし、最小限のノイズと最高の再現性で毎秒 1 回の測定値を出力するコマンド シーケンスを示します。

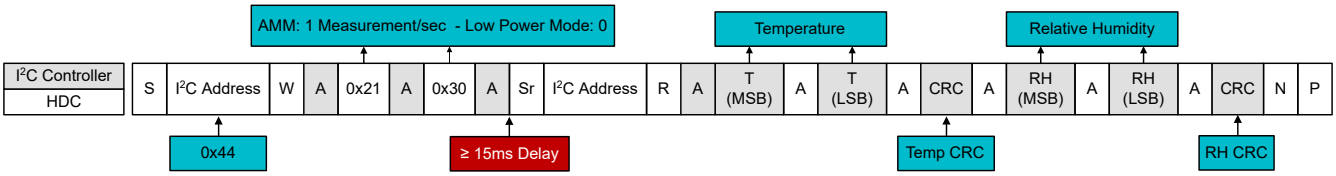

図 2-8. HDC302x 自動測定モード (AMM) 通信構造

```
// configure HDC302x for Auto Measurement Mode (1 measurement/sec)
// lowest noise, highest repeatability
void deviceInit() {

    Wire.beginTransmission(0x44);
    Wire.write(0x21); //send MSB of command
    Wire.write(0x30); //command LSB
    Wire.endTransmission();
    delay(15); //wait 15ms before reading

}
```

デバイスを自動測定モードに構成し、変換が完了するのに十分な時間が経過した後 (この場合は 1 秒) 、以下の機能を使用して、保存されている測定データを要求します。

```
// Helper function for requesting data when in Auto Measurement Mode
void requestData() {

    Wire.beginTransmission(DEVICE_ADDR); // initiate communication
    Wire.write(0xE0); // send MSB of read command
    Wire.write(0x00); // send LSB of read command
    Wire.endTransmission();

}
```

AMM でデバイスを継続的にポーリングするには、READ コマンドを発行して、設定されたサンプリング レート (1 秒あたりの測定値) で最新の測定データを取得します。

こちらの TI GitHub™ リポジトリにある環境センサのセクションで、HDC302x を AMM モードで動作させる方法を示した完全な Arduino サンプル (デバイス構成、測定値のポーリング、データ読み出しを含む) を入手できます。

CRC により測定データを確認する方法

HDC302x の測定出力の CRC チェックを実行することは、特に医療機器、コールドチェーン、気象測定器などの重要なアプリケーションで、データの整合性を確保し、誤測定を防止する上で重要なステップとなります。このセクションでは、HDC302x の既存のコードに即時に実装できるシンプルなコード例を示します。この例では、温度と湿度の測定値の CRC チェックを特に実行することに重点を置いています。

HDC3020 は CRC-8 規格に従います。これは、温度と湿度の測定値に対して固有の 8 ビット CRC 値を出力します。上記の 図 2-8 に、これを示します。ユーザーが湿度データと温度データのどちらをチェックするかに応じて、以下のアルゴリズムは合計 3 バイト、つまり MSB、LSB、CRC をパラメータとして使用し、送信される合計バイト数を受け入れます。次に、データシートに記載されている指定の多項式値 0x31 を使って計算を実行します。式 1 に、CRC チェックの計算に使用される多項式を示します。

$$0x31 \quad = \quad x^8 + x^5 + x^4 + 1 \tag{1}$$

測定データと CRC バイトの両方が処理されると、最終値が 0x00 の場合、CRC チェックが成功し、プログラムは測定読み取り値の出力に進みます。それ以外の場合は、コンソールにエラー メッセージを出力します。

```
// function for checking CRC for HDC measurements
uint8_t checkMeasurementCRC(uint8_t data[], uint8_t dataLength){

uint8_t crc = 0xFF; // initial value per HDC302x datasheet
uint8_t byte;
uint8_t bit;

for (byte =0; byte < dataLength; byte++){ // loops through each byte of input data
    crc ^= data[byte]; // XOR next data byte into current CRC value
    for (bit = 0; bit < 8; bit++){ // process each bit from the data byte
        if (crc & 0x80) // if MSB of CRC is 1
            crc = (crc << 1) ^ 0x31; // shift left and apply polynomial
        else
            crc = (crc << 1); // else shift left, but no polynomial application
    }
}
Serial.print("CRC Check Result: "); // optional; prints CRC value for debugging
Serial.println(crc);
return crc; // return final CRC value
}
```

次に、温度または湿度が読み取られたときにこの関数を呼び出すことで、データの整合性をチェックできます。

```
Humidity Read Example w/ CRC Check:

uint8_thumCheck[3] = {HDC_DATA_BUFF[3], HDC_DATA_BUFF[4], HDC_DATA_BUFF[5]};

// if algorithm output equals final byte value of 0x00, CRC check passes, else output error message
if ((checkMeasurementCRC(humCheck, 3)) == 0x00){

Serial.println("Humidity CRC check passed.");
humidity = getHum(HDC_DATA_BUFF);
Serial.print("Humidity (RH): ");
Serial.print(humidity);
Serial.println("%");

} else {

    Serial.println("Error: Humidity CRC Check Failed.");

}
```

図 2-9 に、出力例を示します。温度データの整合性をチェックする目的で、同じ方法を実装できます。

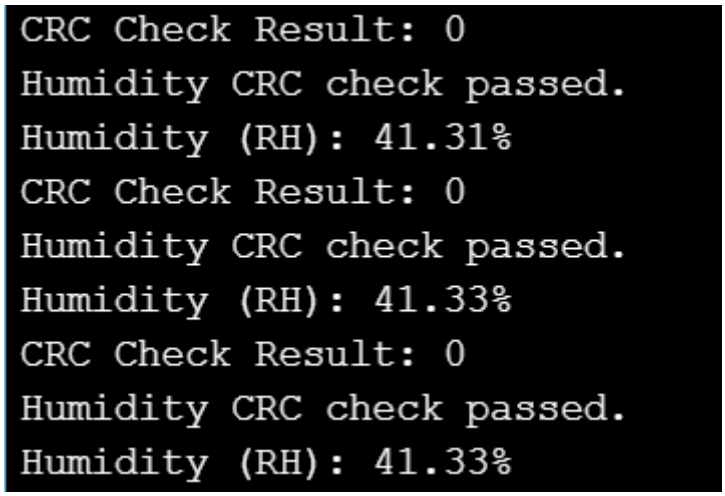

図 2-9. CRC チェックの出力例

C 言語でアラート CRC をチェックする例は、TI の GUI ベースのコード ジェネレータである ASC Studio に記載されています。

3 アナログ インターフェイスの概要

3.1 HDC3120

HDC3120 は、アナログ レシオメトリック出力を搭載したテキサス インスツルメンツ初の湿度センサです。HDC3120 は、温度と相対湿度に対応する連続的な電圧信号を提供するため、低ノイズのアナログ フロント エンド システムに最適な設計です。

通信プロトコルを必要とするデジタル センサとは異なり、HDC3120 では I2C コマンドを書き込んでもレジスタを構成することなく、センサ出力に直接アクセスできます。そのため、A/D コンバータ (ADC) がすでに搭載されているシステムへの統合が簡素化されます。

出力信号を解釈する場合は、ユーザーは適切な変換式 (以下を参照) を適用し、電圧レベルを温度 (°C) と相対湿度 (%RH) に変換することができます。

HDC3120 の重要な特性は、レシオメトリック動作です。温度と湿度の出力電圧は、内部リファレンスとしても機能するデバイスの電源電圧 (VDD) に対して直線的にスケーリングされます。この設計は、電源のノイズとドリフトに対する耐性を備えており、信頼性の高い測定を可能にします。温度と湿度の両方の出力曲線を示す以下のグラフに、この関係を示します。

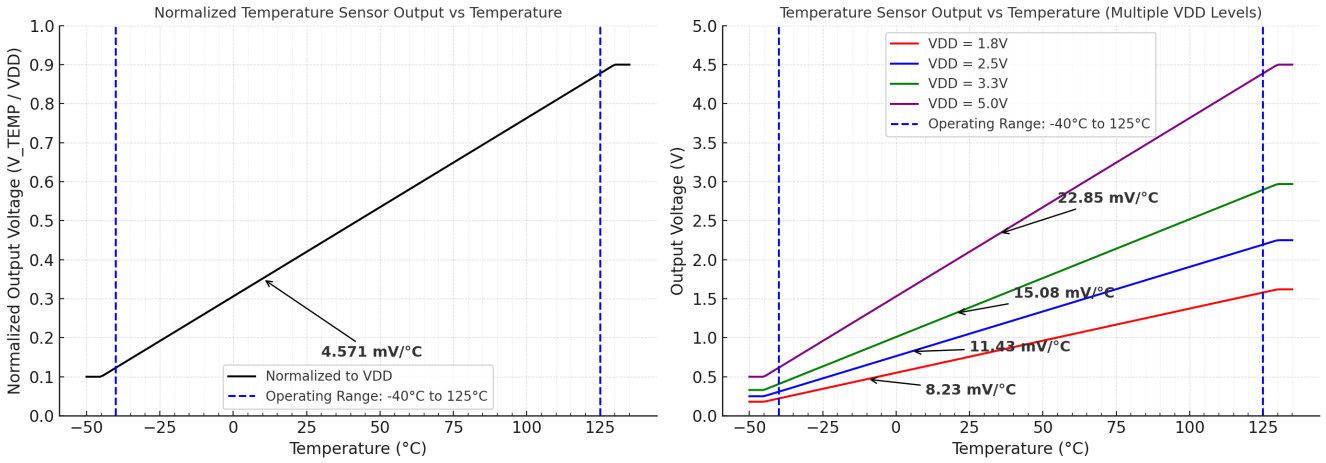

図 3-1. レシオメトリック温度出力プロファイルおよび変換式

$$T(°C) = 218.75 \times \frac{V_{OUT}}{V_{DD}} - 66.875 \quad (2)$$

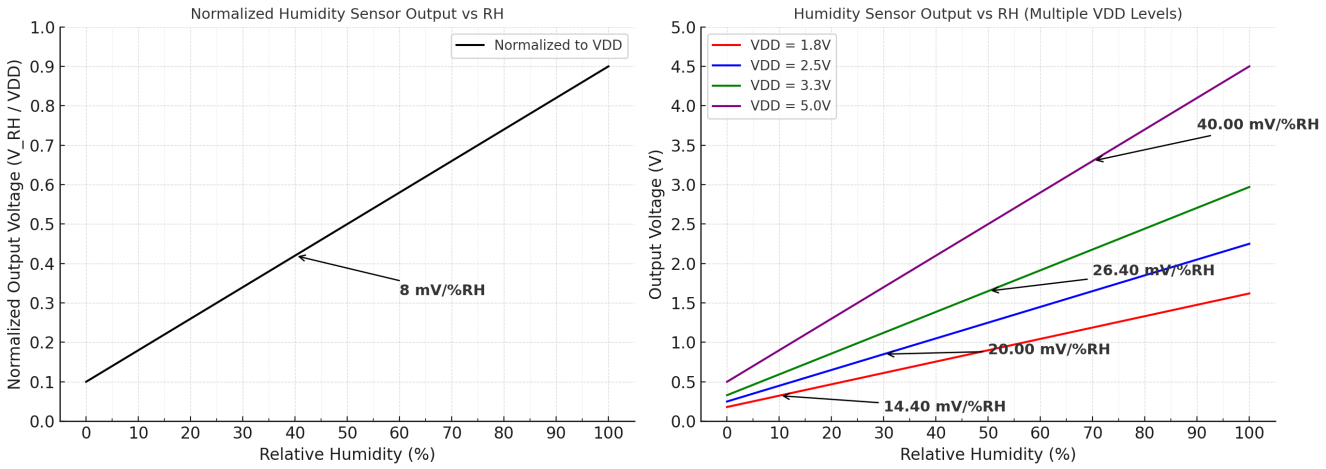

図 3-2. レシオメトリック相対湿度出力プロファイルおよび変換式

$$\%RH = 125 \times \frac{V_{OUT}}{V_{DD}} - 12.3 \quad (3)$$

特に ADC と接続する場合に、HDC3120 出力のレシオメトリック特性を理解することが重要となります。これは、センサの出力電圧が電源電圧 (VDD) に直接スケーリングされるためです。選択した ADC が電源もリファレンスとして使用する場合、電源電圧内にノイズまたはドリフトが発生しても、測定の一貫性が維持されます。この整列によって、ゲインの不一致の問題が解消され、センサの測定値の一貫性を維持することができます。

HDC3120 に適した ADC の選択方法:

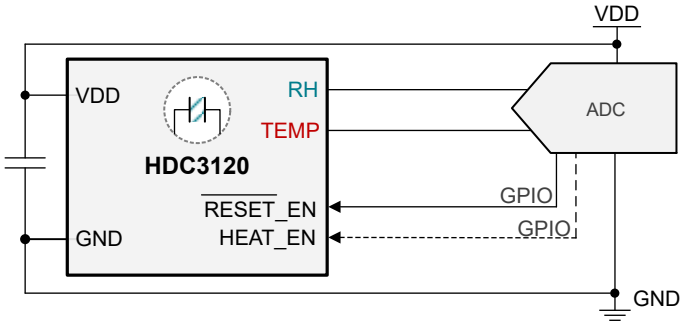

図 3-3. HDC3120 から ADC へ

表 3-1 に、一部の HDC3120 向け ADC を示します。

表 3-1. HDC3120 向けの ADC

ADC	分解能	レシオメトリック	車載 グレード	電源電圧範囲	選択するタイミング
ADS7142	12 ビット	あり	Q100	1.65～3.6V	システム制限超過のアラート機能内蔵。最大 2 つのシングルエンド入力
ADS7138	12 ビット	あり	Q100	2.35V～5.5 V	GPIO ピンによるヒーター/イネーブル制御。最大 8 つのシングルエンド入力
ADS7066	16 ビット	あり	—	3V-5.5V	GPIO ピンによるヒーター/イネーブル制御。最大 8 つのシングルエンド入力
ADS1015	12 ビット	なし	Q100	2V-5.5V	プログラマブル ゲイン アンプ (PGA) により、より広範な入力電圧範囲が実現。最大 4 つのシングルエンド入力

HDC3120 の ADC を選択する場合は、以下のプロセスを考慮する必要があります。

1. ADC の最下位ビット (LSB) を決定する
2. HDC3120 の温度 LSB を計算する
3. HDC3120 の湿度 LSB を計算する
4. LSB 値を比較する

HDC3120 の温度 LSB よりも LSB が低い ADC を選択します。湿度 LSB にも同じ考慮事項を使用できますが、HDC3120 の温度センサの精度は高いため、ADC に必要な最小 LSB を決定するために使用されます。

ADS1015 には内部リファレンス電圧しかなく、外部リファレンス電圧を使用できないため、レシオメトリック測定を実行できません。また、以下の式を使用して LSB を求める必要があります。

$$LSB = \frac{FSR}{2^n}; \quad FSR_{ADS1015} = \frac{2 \times V_{REF}}{gain} \tag{4}$$

ここでは、FSR (フルスケール レンジ) はスケーリング係数を表し、。V_{REF} は、ADC リファレンス電圧です。ADS1015 の場合、ADS1015 データシートの表 7-1 を使用してプログラマブル ゲイン アンプ (PGA) を設定することにより、ゲインが決定されます。

HDC3120 と ADC のペアリングに関する詳細情報については、HDC3120 データシートのセクション 8.2.2 を参照してください。

シナリオの例:

この例では、BOOSTXL-ADS7142-Q1 評価基板を HDC3120EVM と組み合わせています。どちらのデバイスにも同じ 3.3V 電源を使用して電力を供給しています。ここから、上記の手順を使用して両方のデバイスの LSB を比較できます。

1. 以下の式を使用して、ADC の最下位ビット (LSB) を決定します。

$$\frac{FSR}{Resolution} = LSB_{ADC} \tag{5}$$

   a. 動作電圧は 3.3V で、ADC は 12 ビットの分解能を持つため、LSB は以下のようになります。

$$\frac{3.3V}{2^{12}} = 0.805mV \tag{6}$$

2. 以下の式を使用して、HDC3120 の温度 LSB を計算します。

$$V_{TEMP} = V_{DD} \times \left[T(°C) \times 4.571\frac{mV}{°C}\right] \tag{7}$$

$$Temp_{LSB} = 3.3V \times \left[\left(0.1\right) \times 4.571\frac{mV}{°C}\right] \quad = \quad 1.508 \quad mV/°C \tag{8}$$

HDC3120 の標準的な温度検出精度を反映するため、温度 (T(°C)) の値は 0.1 となっています。

3. 式 9 を使用して、HDC3120 の湿度 LSB を計算します。

$$V_{RH} = V_{DD} \times \left[(\%RH) \times 8\frac{mV}{\%RH}\right] \tag{9}$$

$$RH_{LSB} = 3.3V \times \left[(1.0) \times 8\frac{mV}{\%RH}\right] = 26.4 \quad mV/\%RH \tag{10}$$

湿度 (%RH) の値は、HDC3120 の標準的な湿度検出精度を反映するため 1.0 です。

4. LSB 値を比較します
   a. HDC3120 の温度 LSB サイズは 3.3V で 1.508mV です。ADS7142 の LSB は 0.805mV であるため、少なくとも 1°C の測定精度を保持できます。0.5°C など、より高い精度が必要な場合は、ADS7066 などの高分解能 ADC を使用する必要があります。

この設定例では、HDC3120EVM を BOOSTXL-ADS7142-Q1 評価基板に接続しています。両方のデバイスが同じ VDD レールを共有しています。設定手順は以下のとおりです。

1. 図 3-4 に示されているように、ジャンパ ワイヤを使用して、HDC3120EVM から BOOSTXL-ADS7142-Q1 評価基板 BoosterPack™ の AIN0 入力に TEMP 出力を接続します。

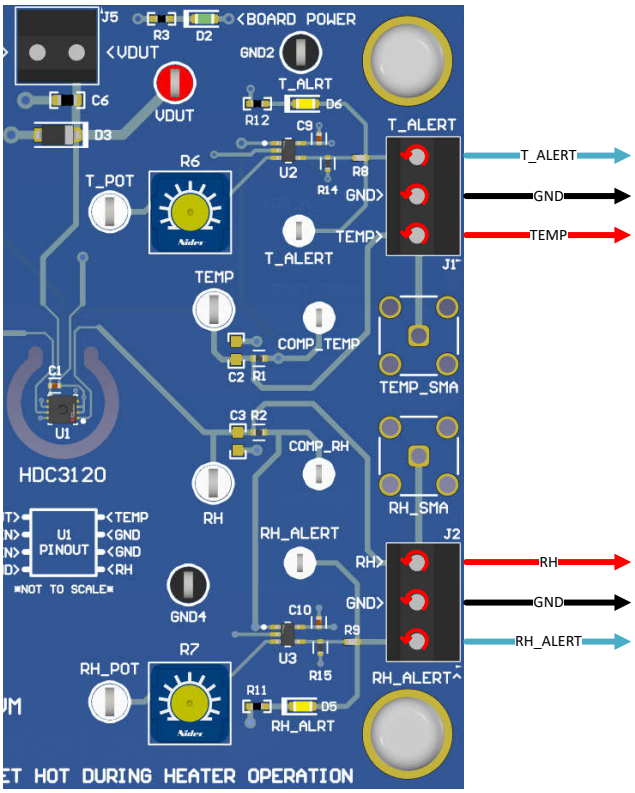

図 3-4. HDC3120EVM 出力

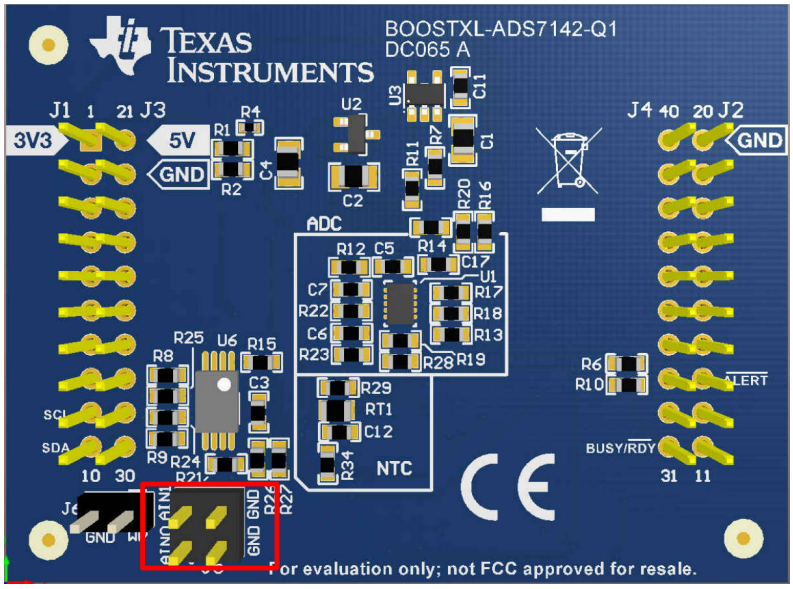

図 3-5. BOOSTXL-ADS7142-Q1 評価基板 BoosterPack™ のピン接続位置

2. HDC3120EVM からの RH 出力を BOOSTXL-ADS7142-Q1 評価基板 BoosterPack™ の AIN1 に接続します。
3. ADS7142EVM GUI を起動し、ホームページに移動します。
4. 構成メニューにアクセスするには、歯車アイコンをクリックし、[変換モード] の下にある [I2C コマンド モード] を選択します。
5. [*動作モード*] ドロップダウン メニューから [*自動 SEQ モード*] を選択して、赤い [*設定*] ボタンをクリックします。
6. 設定が完了したら、[*シーケンスを開始*] をクリックして、12 ビットの 10 進数形式で測定値のキャプチャを開始します。

注

以下に示されている HDC3120 の温度と湿度の出力変換式は、12 ビット BOOSTXL-ADS7142-Q1 EVM 評価基板を使用するアプリケーションに使用します。

結果を解釈するために、HDC3120 の変換式を適用できます。式 11 および 式 12 に、HDC3120 の変換式からの比率の再解釈を示します。式 13 に、変数の追加定義を示します。

$$\frac{V_{RH}}{V_{DD}} = \frac{RH_{ADC}}{2^{12}} \tag{11}$$

$$\frac{V_{TEMP}}{V_{DD}} = \frac{TEMP_{ADC}}{2^{12}} \tag{12}$$

$$\begin{aligned} \boldsymbol{TEMP_{ADC}} &\rightarrow \text{HDC3120 to ADC temperature decimal output} \\ \boldsymbol{RH_{ADC}} &\rightarrow \text{HDC3120 to ADC relative humidity decimal output} \\ \boldsymbol{2^{12}} &\rightarrow \text{represents the 12-bit resolution of the ADC outputs} \end{aligned} \tag{13}$$

注

この例の配線構成では、ピン AIN0 (温度) は GUI のチャネル 0 を表し、ピン AIN1 (湿度) はチャネル 1 を表しています。出力データを受信するため、両方がオンに切り替わっていることを確認してください。

図 3-6 に、HDC3120 の温度と湿度の測定値を示します。チャネル 0 (上のグラフ) は温度測定値、チャネル 1 (下のグラフ) は湿度測定値を表しています。各チャネルの「最新データ」の横に最新データが表示されます。グラフ上の曲線は、HDC3120EVM のヒーター イネーブル スイッチによって HDC3120 ヒーターが 3 秒間有効化され、その後無効化されるまでの約 3 秒間の間隔を表しています。そのため、ヒーターを有効にするたびに温度のピークが観測され、デバイスが加熱から乾燥するにつれて湿気のトラフ (逆のピーク) が観測されています (これは加熱素子を内蔵した湿度センサーでは正常な動作です)。

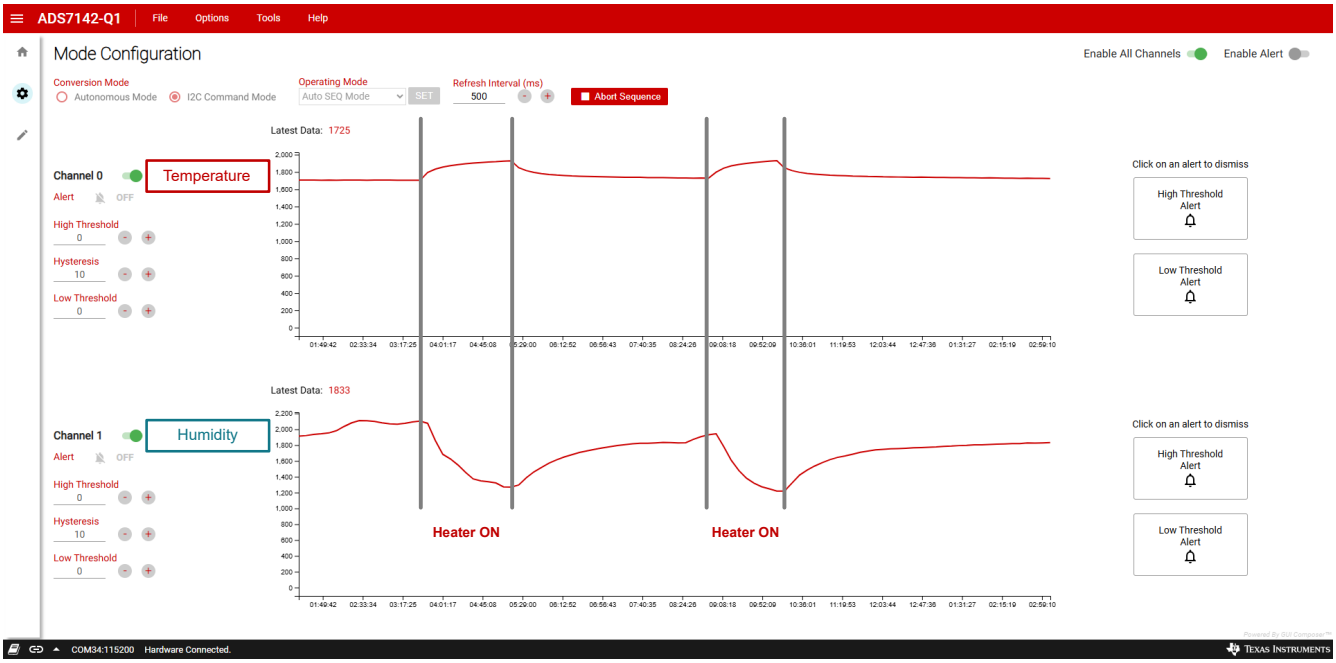

図 3-6. BOOSTXL-ADS7142-Q1 評価基板による HDC3120 の出力解釈

以下の計算では、温度と湿度について最新に収集されたデータ ポイントを使用しています。

温度:

ADC 温度出力 (チャネル 0) :1725

HDC3120 から ADS7142 への温度変換式:

$$\mathrm{T}\left(^{\circ}\mathrm{C}\right) = -66.875 + 218.75 \times \left(\frac{V_{TEMP}}{V_{DD}}\right) = -66.875 + 218.75 \times \left(\frac{TEMP_{ADC}}{2^{12}}\right) \quad (14)$$

$$\mathrm{T}\left(^{\circ}\mathrm{F}\right) = -88.375 + 393.75 \times \left(\frac{V_{TEMP}}{V_{DD}}\right) = -88.375 + 393.75 \times \left(\frac{TEMP_{ADC}}{2^{12}}\right) \quad (15)$$

式 14 を使用すると、変換された温度結果は以下のようになります。25.2°C

湿度:

ADC 湿度出力 (チャネル 1) :1833

HDC3120 から ADS7142 への湿度変換式:

$$\%\,\mathrm{RH} = -12.5 + 125 \times \left(\frac{V_{RH}}{V_{DD}}\right) = -12.5 + 125 \times \left(\frac{RH_{ADC}}{2^{12}}\right) \quad (16)$$

式 16 を使用すると、変換された湿度結果は以下のようになります。43.4 %RH

4 まとめ

テキサス インスツルメンツは、エンド システムに重要な環境情報を提供する湿度センサの製品ファミリを拡充しています。TI の湿度センサを適切にプログラムする方法を理解することで、最新のシステムで最高の性能を実現しやすくなります。湿度センサの各ファミリに公開されているコードは、エンジニアがプロセスを理解して一般的なデバッグの問題を回避する上で役立つ参照を提供することを目的としています。センサの用途やデバイス特性の詳細については、リンク先の資料を参照してください。

5 開発サポートおよび資料

5.1 ソフトウェア サポート:

Arduino™ ベースのコントローラを使用して迅速なプロトタイピングを行うには、まず TI の GitHub™ リポジトリにある環境センサのセクションを参照してください。このリポジトリには、利用可能なすべての湿度センサに対応するサンプル コードが記載されています。

C ベースのドライバレベルのより詳細なサポートについては、まず TI の GUI ベースのコード ジェネレータである ASC Studio を参照してください。

追加のサポートが必要な場合は、TI E2E センサ サポート フォーラムにアクセスしてください。

5.2 参考資料

テキサス インスツルメンツ、『*RH センサにおける RH 精度の問題のデバッグ方法*』、アプリケーション ノート。

- RH 精度の誤差に関する説明と防止策を提供します。

テキサス インスツルメンツ、『*車載エレクトロニクス向け湿度センサベースの水侵入監視*』、アプリケーション ノート。

- HDC3020 センサを利用して、密閉の電子筐体または通気型の電子筐体における水の侵入を迅速に特定する湿度センサベース検出方法を提供します。

テキサス インスツルメンツ、*HDC3x シリコンユーザーガイド*、ユーザー ガイド。

- HDC3x センサ ファミリの保存および取り扱いガイドライン。

テキサス インスツルメンツ、*HDC2x シリコンユーザーガイド*、ユーザー ガイド。

- HDC2x センサ ファミリの保存および取り扱いガイドライン。

テキサス インスツルメンツ、『*温度および湿度センサ向けの NIST トレーサビリティ*』製品概要

- NIST トレーサブル デバイスの特長を説明し、現代的な用途におけるその重要性を解説します。

重要なお知らせと免責事項

テキサス・インスツルメンツは、技術データと信頼性データ (データシートを含みます)、設計リソース (リファレンス デザインを含みます)、アプリケーションや設計に関する各種アドバイス、Web ツール、安全性情報、その他のリソースを、欠陥が存在する可能性のある「現状のまま」提供しており、商品性および特定目的に対する適合性の黙示保証、第三者の知的財産権の非侵害保証を含むいかなる保証も、明示的または黙示的にかかわらず拒否します。

これらのリソースは、テキサス・インスツルメンツ製品を使用する設計の経験を積んだ開発者への提供を意図したものです。(1) お客様のアプリケーションに適した テキサス・インスツルメンツ製品の選定、(2) お客様のアプリケーションの設計、検証、試験、(3) お客様のアプリケーションに該当する各種規格や、その他のあらゆる安全性、セキュリティ、規制、または他の要件への確実な適合に関する責任を、お客様のみが単独で負うものとします。

上記の各種リソースは、予告なく変更される可能性があります。これらのリソースは、リソースで説明されている テキサス・インスツルメンツ製品を使用するアプリケーションの開発の目的でのみ、テキサス・インスツルメンツはその使用をお客様に許諾します。これらのリソースに関して、他の目的で複製することや掲載することは禁止されています。テキサス・インスツルメンツや第三者の知的財産権のライセンスが付与されている訳ではありません。お客様は、これらのリソースを自身で使用した結果発生するあらゆる申し立て、損害、費用、損失、責任について、テキサス・インスツルメンツおよびその代理人を完全に補償するものとし、テキサス・インスツルメンツは一切の責任を拒否します。

テキサス・インスツルメンツの製品は、テキサス・インスツルメンツの販売条件、または ti.com やかかる テキサス・インスツルメンツ製品の関連資料などのいずれかを通じて提供する適用可能な条項の下で提供されています。テキサス・インスツルメンツがこれらのリソースを提供することは、適用されるテキサス・インスツルメンツの保証または他の保証の放棄の拡大や変更を意味するものではありません。

お客様がいかなる追加条項または代替条項を提案した場合でも、テキサス・インスツルメンツはそれらに異議を唱え、拒否します。

郵送先住所:Texas Instruments, Post Office Box 655303, Dallas, Texas 75265

重要なお知らせと免責事項

TI は、技術データと信頼性データ (データシートを含みます)、設計リソース (リファレンス デザインを含みます)、アプリケーションや設計に関する各種アドバイス、Web ツール、安全性情報、その他のリソースを、欠陥が存在する可能性のある「現状のまま」提供しており、商品性および特定目的に対する適合性の黙示保証、第三者の知的財産権の非侵害保証を含むいかなる保証も、明示的または黙示的にかかわらず拒否します。

これらのリソースは、TI 製品を使用する設計の経験を積んだ開発者への提供を意図したものです。(1) お客様のアプリケーションに適した TI 製品の選定、(2) お客様のアプリケーションの設計、検証、試験、(3) お客様のアプリケーションに該当する各種規格や、その他のあらゆる安全性、セキュリティ、規制、または他の要件への確実な適合に関する責任を、お客様のみが単独で負うものとします。

上記の各種リソースは、予告なく変更される可能性があります。これらのリソースは、リソースで説明されている TI 製品を使用するアプリケーションの開発の目的でのみ、TI はその使用をお客様に許諾します。これらのリソースに関して、他の目的で複製することや掲載することは禁止されています。TI や第三者の知的財産権のライセンスが付与されている訳ではありません。お客様は、これらのリソースを自身で使用した結果発生するあらゆる申し立て、損害、費用、損失、責任について、TI およびその代理人を完全に補償するものとし、TI は一切の責任を拒否します。

TI の製品は、TI の販売条件 、 TI の総合的な品質ガイドライン 、 ti.com または TI 製品などに関連して提供される他の適用条件に従い提供されます。TI がこれらのリソースを提供することは、適用される TI の保証または他の保証の放棄の拡大や変更を意味するものではありません。 TI がカスタム、またはカスタマー仕様として明示的に指定していない限り、TI の製品は標準的なカタログに掲載される汎用機器です。

お客様がいかなる追加条項または代替条項を提案する場合も、TI はそれらに異議を唱え、拒否します。

最終更新日：2025 年 10 月